

Solutions to Practice Problems for Homework 4

1. Your implementation is to be designed for the following conditions. There are 10^8 phone numbers that will be inserted with 10^5 different area code/exchange (first 6 digits of the number) combinations occurring. Further, suppose you only have space in main memory to hold 8,000,000 bytes. (An integer takes 4 bytes (32 bits) and a long takes 8 bytes.)

The primary data structure we'll use is a mapping M implemented using open addressing. The elements stored in the mapping are tagged elements in which the tag is the area code/exchange pair (4 bytes), and the associated data is the page location for a secondary mapping (8 bytes) that will maintain the needed data for all phone numbers with that area code and exchange.

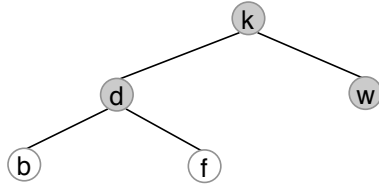
For the primary mapping M , we'll use open addressing since there is no deletion required and by using open addressing we can fit the entire hash table in main memory. We must now determine an appropriate size for the hash table. Let's suppose that we only want to use half of memory (4,000,000 bytes) for the primary hash table so that there is space in memory to bring in the secondary mapping for a desired area code. Since $\lfloor 4000000/12 \rfloor = 333333$ a good size for the hash table is 262144 which is a power of 2 and so we can use the primary and secondary hash functions used from lab 2. Hence the load factor $\alpha = 100000/262144 \approx .38$. With this the expected number of probes in an unsuccessful search is $1/(1 - \alpha) \approx 1.6$.

The tag for the secondary mappings will be the last 4 digits of the phone number, and the associated data will be the name and address associated with that phone number. While we know that there are an average of $10^8/10^5 = 1000$ phone numbers in each of these, there could be as many as 10,000 entries in some. Hence for the secondary mappings, separate chaining will be selected with a hash table size of 1024. (In the worst case of 10,000 entries in the table, we would expect each list to have under 10 items which can still be searched quickly. Overall, there is just an expectation of just under one item per list.)

We now describe how the required methods will be implemented.

- Insert a new area code/exchange combination into the system. (This will be used infrequently).
First an insertion is made into M with the given area code/exchange as the key. Then we create a new (empty) secondary mapping for the given area code/exchange. This will take constant time (we expected to make 1.6 probes for the insert). In addition, the secondary hash table must be initialized, and the page needs to be written to disk.
- Insert into the system a new item that consists of a phone number, address, and name. (You can assume that the area code/exchange from the number corresponds to one of the area code/exchanges already in the system.)
First the primary mapping M is used to find the location of the page with the secondary mapping for the given area/code exchange pair. This step has constant expected time to find the location of the page. Then in constant time the page can be read into memory. (The 4,000,000 bytes that are still available in main memory is large enough to hold the page). Then using the last 4 digits as the tag, we can insert the new phone number into the secondary mapping. This step has constant expected time. Finally, the page can be written back to disk in constant time. Hence overall this has expected constant time complexity.
- Given a phone number, return the address and name.
First the primary mapping M is used to find the page with the secondary mapping for the given area/code exchange pair. (Then that secondary mapping data structure can be brought into memory as discussed above). Then using the last 4 digits as the tag, we can retrieve the name from the secondary mapping. Each step takes constant expected time.
- Given a phone number, remove the corresponding item from the system.
This is just like the above, but remove is called on the secondary mapping. Thus it takes constant expected time.

2. Recall that in each red-black tree the filled nodes are black and the unfilled nodes are red. The initial tree given was:



(a) The black height is 2.

